Analysis and Improvements in Trojan Designing

A Thesis Submitted on
14th May, 2012
in partial fulfillment of the requirements for the degree of

Bachelor of Technology
in
Computer Science and Engineering
by

Shanti Swaroop Swain
(Roll No-108CS003)

Under the Guidance of
Prof. S. Chinara

Computer Science and Engineering
National Institute of Technology, Rourkela
Rourkela – 769 008, Odisha, India

Computer Science and Engineering
National Institute of Technology, Rourkela
Rourkela – 769 008, Odisha, India

Certificate

This is to certify that the work in the thesis entitled ‘Analysis and Improvements in Trojan Designing’ submitted by Shanti Swaroop Swain is an original research work carried out by him under my supervision and guidance for partial fulfillment of the requirements for the award of the degree of Bachelor of Technology in Computer Science and Engineering during the session 2011-12 in Department of Computer Science and Engineering, NIT Rourkela.

To the best of my knowledge, the matter embodied in this report has not been submitted to any other University/Institute.

Date -14/05/2012

Rourkela

(S. Chinara)

Department of Computer Science and Engineering

NIT Rourkela

Acknowledgments

I express my profound gratitude and indebtedness to **Prof. S. Chinara**, Department of Computer Science and Engineering, National Institute of Technology, Rourkela, for introducing the present topic and for her inspiring intellectual guidance, constructive criticism and valuable suggestions throughout my project.

I also express my sincere gratitude to *Prof. A.K.Turuk*, Head of the Department, Computer Science Engineering, for providing valuable departmental facilities. In addition to this, I would like to thank *Prof. S.K.Jena*, *Prof. B.D. Sahoo*, *Prof.D.P.Mohapatra*, *Prof. K.S. Babu* for their time to provide more insightful opinions into my research.

I do acknowledge the academic resources that I have received from NIT Rourkela. I also thank the administrative and technical staff members of the Computer Science Department for their in-time support. I would also like to thank my parents and my friends, for their cooperation and motivation for completion of this project.

Date-14/05/2012

Rourkela

Shanti Swaroop Swain

Abstract

While there are numerous Trojans out there in the internet, all of them are easily detected by antiviruses or blocked by firewalls. Apart from that, they are also easily detected and removed by any user with a good knowledge in Operating System and Security. This project's objective is, therefore, to identify and remove the design flaws, add some improvements and features to make it undetectable. Antiviruses identify a threat mostly based on two factors. Either signature matching or heuristic analysis based of certain suspicious behaviors and patterns. This project here doesn't consider the *Blacklisting* feature of an Anti-Virus. Now to avoid detection based on the other two factors, the main objective is to make the Trojan *look* like a normal legal program. To achieve this, the best way is to use the legal and secure facilities provided by Windows itself. This way, the Trojan will install and work similar to any other software; however, everything will be done stealthily. Apart from the traditional objective of *giving backdoor access to victim's computer*, this Trojan here includes another objective of *Bypassing firewalls and protecting itself for functioning properly* as well. There is a tradeoff too and that needs to be mentioned before we proceed further. This tradeoff is between size and detection. The technique that is applied here to make this Trojan perfect will certainly increase its size. While traditional Trojans are less than 50KB in size, the proposed Trojan will be more than 400KB. Still, it isn't a great problem as long as it is able to stay hidden.

Table of Contents

List of Figures

Chapter 1: Introduction

A Trojan horse, or Trojan, is a program with a benign capability that conceals another malicious program. When the user executes a Trojan horse, the program performs the expected task; however, the program is also performing actions unknown to, and not in the best interests of the user.

This program is used for spying and some other illegal purposes. The objective of the Trojan is to secretly get into the victim's computer and install itself. By doing this, Trojan now can give a backdoor access to the Hacker to this hacked computer. Hacker can do anything he wants with the victim's computer with full access rights.

Some purposes of Trojans are, to log whatever the user is typing, starting for passwords to private chats; to capture screenshots and view whatever the user is doing in his/her PC. You are watching a private picture secretly!! Don't be so sure, hacker is watching too; to see what files you have, to download some private pictures of yours, or may be some confidential employer information. Your documents won't be private anymore. Trojans are also used to steal Credit Card information, your browsing history, your private E-mails and many more. It can also be used to spy on you, using your Webcam or your Microphone. **Are you sure no one is watching you when you are alone in your room??? Or maybe you *think* you are having a *private* conversation with your best friend.**

This is what a Trojan is. Once you are a victim of a Trojan, you don't have a chance to protect your privacy. Many people confuse Trojans with Viruses. In fact, there is a great difference between the two. Objective of a Virus is to destroy your computer, to corrupt your files, to cause you harm and make you suffer. If a Virus destroys an Operating System and its files, it is considered to be a great success. In other hand, Trojan's objective is to stay hidden and cause no visible trouble to the user at all. Because, more silent it stays, longer it will be able to help the Hacker steal information. If a Trojan causes any visible trouble to the user, it is considered as a failure.

Now it's apparent how dangerous this program is. It's actually the most dangerous in the world of hacking. **A virus might just destroy your files, but a Trojan can destroy you.** Every complex hacking method almost always ends up with backdooring. A Hacker does get complete control over a computer after a serious hacking procedure; however to save time next time, he uses a Trojan. So that next time he could get similar level of access without performing the hacking again.

Trojans are not only used by hackers; these are also used by government organizations and private companies. Government organizations like FBI, uses this to spy on others. Private companies use this to spy on employees of other companies to gain confidential ideas or information that might help them compete and win.

Prior to the development process, it is necessary to understand one basic thing about Trojans. How it enters into your computer? There are 2 ways. First, a Hacker might hack your computer by exploiting some software vulnerabilities and then leave a Trojan in your system. And second, you might be fooled into executing a Trojan yourself. In both the cases, the Trojan design is exactly same. However in the second case, there are some extra things to be done.

In this project here, the focus is on the second method. This method is the most prevalent method of Trojan usage. What happens here is, Hacker injects a Trojan into an unsuspicious file like a software setup file, or some game setup files or may be cracks or patches. Once injection is done, the infected file is modified to create an illusion of legitimacy. This file is then given to a user or may be shared over the internet.

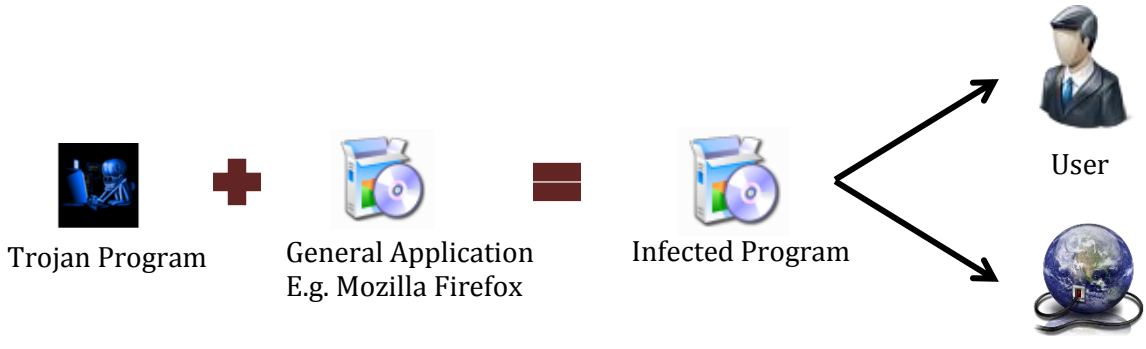


Fig. 1: How Trojan is used

When the user executes the file (double clicks the infected file), the file splits into two parts: The Trojan part and the original file. Both are executed differently. Original file is executed normally so that the user gets what he was expecting. Trojan program is executed in hidden mode. This program runs in the background and the user is completely oblivious about this. Understand that this splitting of the file and executing the parts in different modes are done by another program called Extractor. This Extractor is also added to the Original File in addition to the Trojan specifically for this purpose.

Chapter 2: Flaws in the Traditional Design Process

- **Assembly Language Programming**
  - Assembly language is the most preferred language for designing Viruses and Trojans. It makes a Trojan very light weight and gives low level access to the system. However the coding pattern remains almost same for all Trojan Designs. Now if an Anti-Virus uses Pattern Matching techniques or Signature Matching for detection, and a Trojan gets detected, then almost all other Trojans with similar functions will get detected as well.

```
.386P
Locals
jumps
.Model Flat ,StdCall
mb_ok     equ 0
hWnd      equ 0
lpText    equ offset text
lpCaption equ offset caption
extrn     ExitProcess     : PROC
extrn     MessageBoxA     : PROC
```

```
.Data
text     db "Hello World",13,10 ;
caption  db "Hello",0 ;
.Code
Main:
      push mb_ok;
      push lpCaption ;
      push lpText   ;
      push hWnd     ;
      call MessageBoxA  ;
      CALL    ExitProcess  ;
End Main
```

This example code is of “Hello World” program in assembly language. Now anyone in the world may write this code, and there is a great chance that most of their codes will be similar to this. Assume that this code is a malicious code. Then Anti-Viruses will obviously detect any Trojan that has same syntax structure as this and consider it as a threat.

- **No Modularity**
  - In traditional Trojans a single assembly program (or module) includes all the functions for all purposes. So when a single function or activity is detected as malicious, whole program is branded as a threat and is removed. Because of that single function, the Trojan is unable to protect itself.

- **Fixed Properties.**
    - A Trojan generally has the same property in every computer. For E.g. Installed Folder, Port Number, Registry Entries, etc. These properties are set by the Trojan developer. But the flaw is that, once a Trojan is detected in any computer, it becomes useless in other computers too. For example, 31337 is used as port number in a Trojan program called Back-Orifice. This is known. So every firewall by-default blocks it.

- **Unintelligent download logic.**
    - Certain Trojan actually doesn't contain any malicious code. These Trojans are simply downloaders. These downloaders download the Trojan Payload from the server and execute it. Payloads are simply exe files that are downloaded in the background. This process is considered malicious. Anti-Viruses brands any process as suspicious, if it is a console program, with hidden window, running as user process, and trying to download an exe in the background.

- **Blocked by NAT servers.**
    - Trojans works on client-server model where a connection needs to be established by the client to the server before any communication. But this is not possible if a NAT server exists in the network. NAT server blocks any connection originated from outside the network. So if the Hacker is outside the network, he cannot connect to the victim inside the network.

These were the few examples of flaws because of which, such Trojans are easily detected and blocked or removed. To develop a better Trojan, these flaws must be considered and some efforts must be given to improve them. Addition to this, few more features needs to be added to make this Trojan robust and fail-proof.

Before working out the improvements, it's essential to understand something else. Its known that key loggers are detected and branded as threats by Anti-Viruses. But there are many types of software which actually depend upon key logging to function properly. For example, many

hotkey programs used by gamers use key logging techniques. Now the question comes, why are they not branded as key loggers? If anyone wonders on this questions and tries to understand the difference between a keylogger and this hotkey programs, he will certainly find a great difference. Anyone with this knowledge, if designs a keylogger, will be able to bypass every detection mechanisms as there won't be any difference between genuine programs and your keylogger. As its not legal to disclose the programming technique of a keylogger or any malicious functions of a Trojan, the codes are omitted from this thesis. But it's not very difficult once you understand the logic behind it.

The point of writing the last paragraph was that if a Trojan is made to work like genuine windows software, it could easily avoid detections. There is a great difference between general softwares and a traditional Trojan, but if this difference could be minimized then chances that any Anti-Virus would brand it as threat minimizes too.

Chapter 3: Proposed Improvements

- **Using High Level Language.**

  C, C++ lets the IDE create the assembly code which is generally very different from user designed assembly code. So detection based on pattern matching fails. Now the same “Hello World” program in assembly code would look very different. Here is the example of the same:

```
#include <windows.h>
int main()
{
008713A0  push      ebp
008713A1  mov       ebp,esp
008713A3  sub       esp,0C0h
008713A9  push      ebx
008713AA  push      esi
008713AB  push      edi
008713AC  lea       edi,[ebp-0C0h]
008713B2  mov       ecx,30h
008713B7  mov       eax,0CCCCCCCCh
008713BC  rep stos    dword ptr es:[edi]
          MessageBox(NULL,L"Hello World",L"Win32 Assembly",MB_OK);
008713BE  mov       esi,esp
008713C0  push      0
008713C2  push      offset string L"Win32 Assembly" (875758h)
008713C7  push      offset string L"Hello World" (87573Ch)
008713CC  push      0
008713CE  call      dword ptr [__imp__MessageBoxW@16 (878338h)]
008713D4  cmp       esi,esp
008713D6  call      @ILT+310(__RTC_CheckEsp) (87113Bh)
          return 0;
008713DB  xor       eax,eax
}
008713DD  pop       edi
008713DE  pop       esi
008713DF  pop       ebx
008713E0  add       esp,0C0h
008713E6  cmp       ebp,esp
008713E8  call      @ILT+310(__RTC_CheckEsp) (87113Bh)
008713ED  mov       esp,ebp
008713EF  pop       ebp
008713F0  ret
```

- **Splitting the Trojan into modules.**

  Trojan program is divided into modules based on certain criteria like

  1. Suspicious & Unsuspicious Activities
  2. Stable & Unstable Functions
  3. Frequency of Usage

Splitting the Trojan into modules has its advantages. First and foremost is that in case one module gets detected as a threat, only that module is removed. Other modules remain and perform their tasks. This makes a Trojan somewhat robust. Apart from this, dividing modules based on stability is also a good programming skill. If at some point of time, a module crashes due to any reason, the whole program is not terminated. Modularizing based on the frequency of usage, makes the core Trojan program small in size. This way, the Trojan payload has only those modules or codes that it needs frequently. For any other activities, it can download the module and execute it.

- **Variable Installation Properties**

  As mentioned in Chapter 2, fixed properties make a Trojan highly unreliable. So this Trojan is developed in such a way that every time the Trojan installs in a PC, it changes its installation settings like, Location, Registry key entry, Port number, File name, etc. This way, in case it gets detected in one PC, the properties are not same in other computers, so they remain hidden there.

- **Using HTTP Protocol for Communication with Server**

  Trojan needs to communicate with our server in order to receive updates and other commands for its working. However using some unauthorized port number for communication might trigger suspicion. Again, in some strict networks, all ports are blocked except HTTP ports for browsing. So in such networks, it's necessary to use these ports for communication. Trojan can be made to impersonate a browser by sending fake header information. This way, it becomes very difficult for a packet filter or a firewall to determine if the packets are from a Trojan or a browser.

cURL library for C is used in this project in order to achieve HTTP based communication.

- **Using Backup Server**

  Trojan communicates with a primary server. However, a backup server is also set in it so that it can fall back to this backup server incase if the primary server is blocked.

- **Reverse Connect to Hacker**

  Presence of a NAT server in the network will block every attempt from Hacker to connect to the victim behind the NAT. However connections originated from inside the network are allowed. So this concept is used to add a feature in this Trojan to connect back to the Hacker. This way, Hacker doesn't have to connect from outside.

- **Download Trojan as Character Stream**

  Trojan Downloader as a hidden process cannot download an .exe file in the background. But here, the .exe file is converted to normal text file. This text file is uploaded to the server. Now this Trojan downloader downloads the program as text file and again converts back to binary file. Now the downloader is not downloading any executable file in the background, so it's not detected.

- **Running as System Service**

  Windows Service Manager is an internal implementation in Windows where every program that needs to be started with windows for its functioning are registered. Any program registered in System Service Manager, starts just after Windows is booted and before user logs in. So this facility is used too. Traditionally Trojans are entered into registry to run at startup. However in such case, Trojan runs with user access right. But the proposed Trojan here is installed as a System Service. When started, it gains System rights. As by default, all service programs are invisible, it is not required to hide any window. Now this Trojan is running invisibly and with System rights.

- **Avoid Direct Access to System Resources**

  Trojans written in assembly language usually performs every action by accessing the resources directly. Using System call APIs by the operating system to perform all actions might be a little slower than direct access, but it's less suspicious. It also makes the Trojan seem like some other normal software built for windows.

- **Adding resources into the program makes it look more genuine to users**

  Adding resources like, manifests, copyright, file description improves the camouflage of the program making it seem like a normal windows program. Even the File access times, modified time and other details are copied to make the file look legitimate.

These were the few major changes done to the traditional Trojan designs. Apart from this a number of minor modifications were done too. Certain examples include,

- Changes to Programming Logic and Infection Technique.
- Logic for gaining admin rights as well as system rights, etc.

Chapter 4: Designing Trojan

Before finding out how to create a Trojan, it's better to know the technical definition of a Trojan. Knowing this definition will complete 50% of the design logic.

In technical terms, Trojan is a simple client-server program or also called as Remote Access Software that works on Command-Response basis. In simpler words, Trojan is a Client-Server software. Server is installed in the victim's computer and Client is with the Hacker. Hacker sends a command to the Server program, the program performs the appropriate action and the result or information is sent back to the Hacker's Client program.

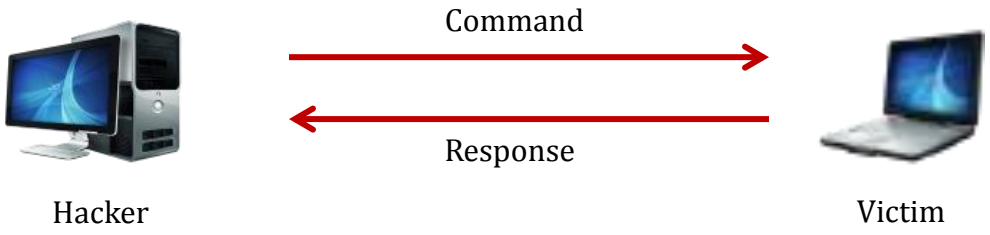


Fig. 2: How Trojan Works

So basically there are two modules: a Server and a Client. As in any client-server programming, this Server program also binds to a port number and starts listening for connection from Client. When Client connects to it, the Server program accepts the connection and starts a separate thread for it. This thread receives commands from the Client and performs the required task.

However, this above mentioned steps will only give you the core of the Trojan program. The core, that will give complete access to the victim's computer. But this core independently is not so effective. It is easy to detect and not so robust. So generally, apart from this Server module (also called as Payload), Trojans have some extra codes in them to keep this server module hidden. Codes are also present for other tasks like infecting other programs, installing itself, etc.

All these points as well as the proposed improvements are taken into consideration while designing this Trojan. And based on the modularizing technique explained in chapter 3, Trojan software is divided into 6 modules.

- Extractor
- WatchDog
- Updater
- Tracker
- Payload
- Injector

Extractor and Injector are not the core modules of our Trojan, but still they play a crucial role in any Trojan development. Here is the description of each module.

- Injector:

  This program is used to create an infected file from a normal file. It takes the Trojan modules and wraps them around the original file. It also copies all the properties and attributes of the original file to this new infected file to make it look exactly similar and unsuspicious.

- Extractor:

  This program is at the top of the infected file. When victim double clicks this infected file, Extractor being on the top, executes first. This program then unwraps all other Trojan modules from the original file and executes the original program. It then copies the modules to appropriate locations, does some things to hide the modules and then executes the main module in hidden mode.

- WatchDog:

  This program is the main module that is started by Extractor. It is solely responsible for the functioning and stability of all other modules. Once this module starts, it starts the next three modules and keeps on checking their status. In case any other module is terminated for any reason, this module starts it again. WatchDog is installed as a system service and is started automatically when windows starts. It gains System right and passes on this access right to other modules.

- Tracker:

  Tracker module is runs for a very short period of time. Its job is to collect certain system information and send it to the server for registration. This registration system is responsible for keeping a track on the victim. No matter how many times the IP address changes, it will always be updated to the server.

- Updater:

  Updater's objective is to keep checking for updates in the server. If any update is available, it downloads it and replaces the existing modules with the updated modules. This way, Hacker is able to add more functionality to the Trojan, replace buggy codes, or update it for any reason without any need for physical access.

- Payload:

  This module is the Server core of the Trojan. This module receives commands from the Client and performs its activities. It is started by WatchDog with System rights. It includes various ways by which a Hacker can connect to it. Once connection is established, the Hacker gains complete access to the computer.

4.1: Designing Injector:

This program is responsible for infecting a normal unsuspicious program with Trojan modules. The infection is actually not a very appropriate word technically. The correct word would be "wrap". This program simply wraps the original program with the modules. This wrapping is done in a very different way.

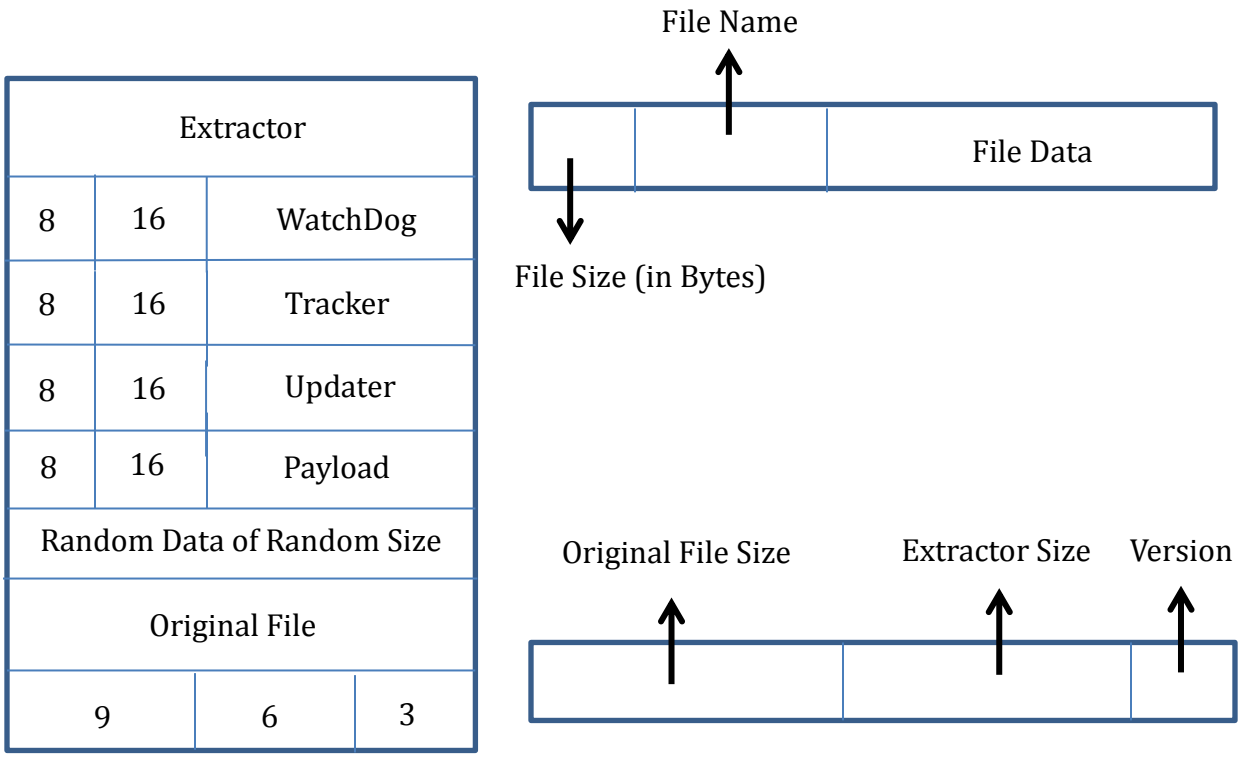


Fig. 3: Infected File Format. (All sizes are in Bytes)

Version is for Hacker's use only. It differentiates an infected file from a clean file. Random data is added to the format to create a random size increment difference between 2 different infected files. Icons and File Version Info are considered as Resources in Windows programming. So programming is done to copy these Resources from original file to the Extractor file. As Extractor file is the first executable module in the file, its icons and info are displayed in windows. After copying the resources our file seems like the original file with exactly same icon image and version information. Apart from this, file creation date, last modified date, and last access date are also copied. *Note: Digital Signatures cannot be copied*

4.2: Designing Extractor:

This program simply extracts the modules from the infected file. It's easier to develop and is complete dependent on the format shown in Fig. 1.

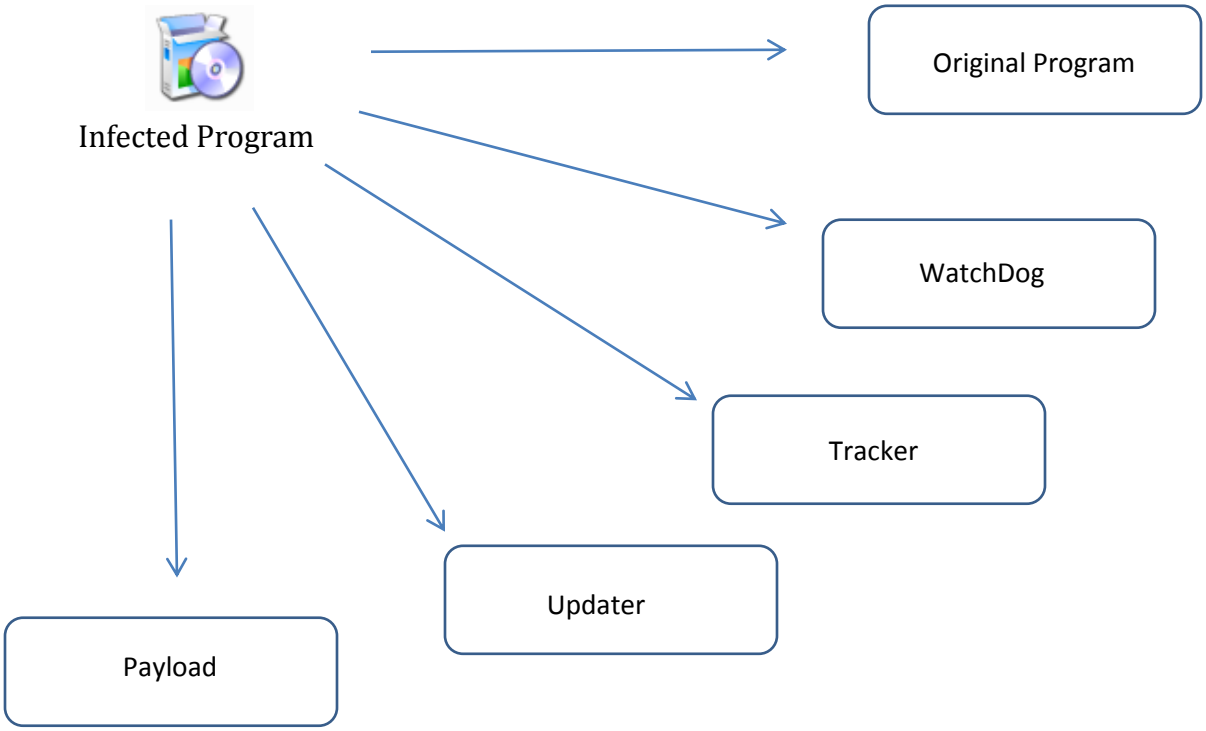


Fig. 4: Extractor splits the file into sub files

Once the files are extracted to a temporary location, Original File is executed in normal mode. So the user finds what he was expecting. This Extractor also receives any command line parameter used to execute it and passes on to the original program. Now the other modules are copied to some secret location in the Victim's computer and are made hidden by changing the file attributes.

Once this hiding process is completed, Extractor checks if Trojan is already installed in the PC. If yes, then it checks for its status. If everything is found to be correct, it deletes all the modules it copied just now and terminates. If no previous installation is found, the main module i.e. WatchDog module is executed with "install" parameter and Extractor terminates.

4.3: Designing WatchDog:

WatchDog is the module that is responsible to *keep an eye* on the proper functioning of other modules. This module is started by the Extractor module with "install" parameter. Once started, it installs itself into Windows Service Manager so that it could start directly every time when Windows starts.

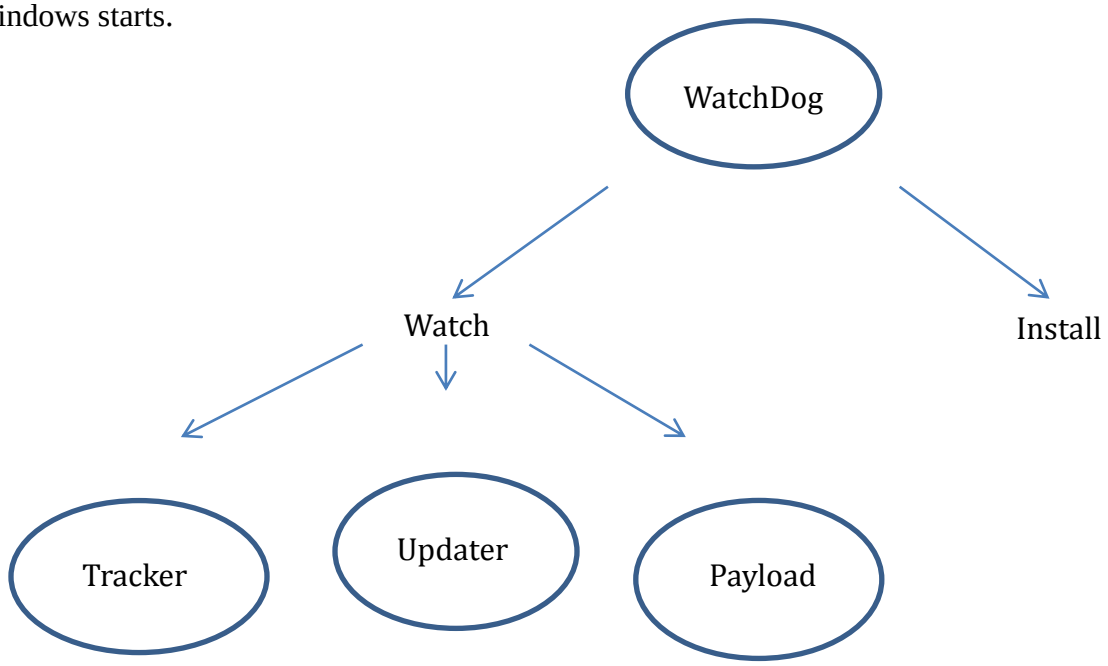


Fig. 5: WatchDog Service Workflow

Next time when Windows starts, WatchDog module is started without any parameter. Now *Watch* function is called. This function starts 3 threads. Each thread starts a new process and keeps on checking their exit codes. All the modules are hardcoded to return a specific return value when terminated successfully. If any forceful termination occurs, then the return code is not the one specified. So WatchDog starts the process again. This way, there is no way anyone can terminate a process while WatchDog module is running unless the process module is deleted. However that is also not possible while the program is running.

WatchDog is started by the Service Manager, so by default it gains System Rights. This access right is passed on to other modules. So other modules are also started with System rights.

4.4: Designing Tracker:

This module keeps the Hacker updated with the system details on the victim's computer. It is started by WatchDog, and once started; it collects little information about the system like, Username, Computer Name, IP address, Mac address, etc. This information is updated in the server database to keep a track of all the Trojan victims.

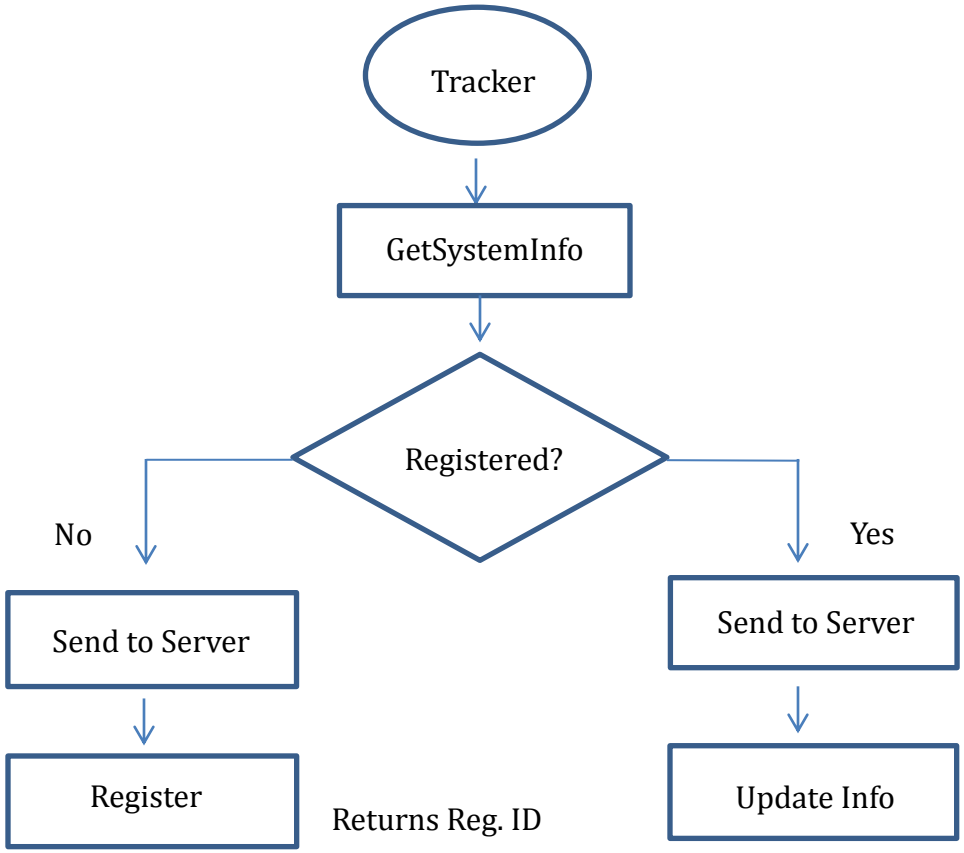


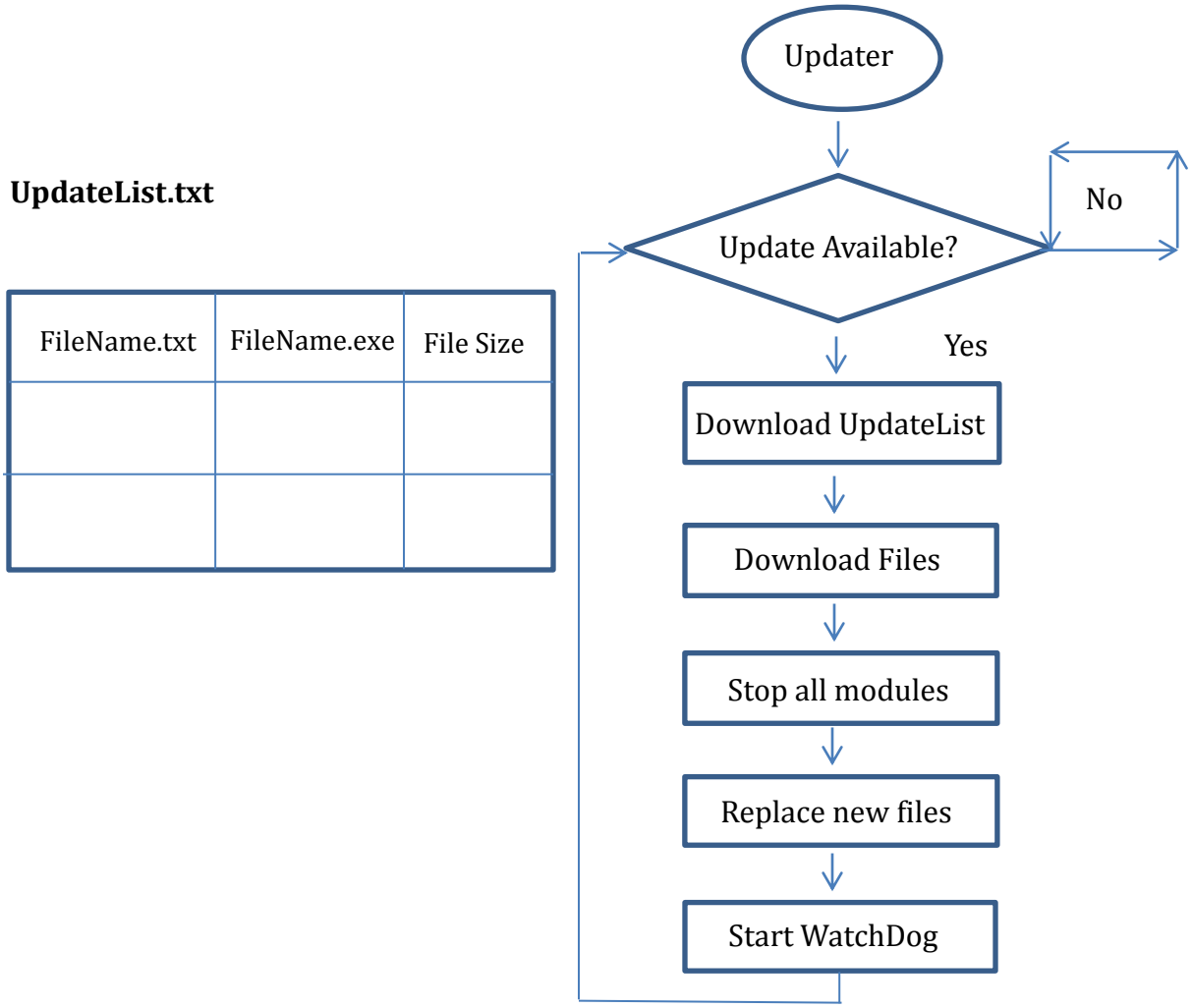
Fig. 6: Tracker Module Workflow

Tracker connects to the server and registers the computer for a record. If the computer is already registered, then any changes to the system information are updated. Upon registration, server returns a registration ID, which is used by other modules for further communications.

Curl library for C is used to implement HTTP protocol based communications between the Trojan and the Server. Because of this, the data is transferred as normal http data and uses port 80 or 443. For more security, the Trojan impersonates a web browser by sending fake header information. This will fool the network administrator if he decides to sniff on the network traffic to detect any malicious packets. It will also bypass most firewalls.

4.5: Designing Updater:

The objective of this program is to enable the hacker to update the Trojan in any victim computer without any physical access. This way, he can add more features to the Trojan and modify the properties and functions for correct functioning of all the modules.



UpdateList.txt

FileName.txt	FileName.exe	File Size

Fig. 7: Updater Module Workflow

All updated files are first converted to text files. These text files are uploaded to the server. Updater downloads these text files and converts back to exe files. It then stops WatchDog and other modules, replaces the files with new files and starts WatchDog module again.

The update list is a text file where all the new file names are entered. First column is the .txt file it has to download from server. Second column is the filename that the text file after conversion will be renamed to. Third column is the actual size of the .exe file and is use for error checking.

4.6: Designing Payload:

Payload is the core module of our Trojan which has all the functions for computer control. This is a server program. It is started by WatchDog and runs with System right. Initially, it binds to a port number and listens on it. It has few other modes with which it can communicate with the Hacker. Objective of this module is to receive commands from the Hacker and process it. Perform as it is required to, and then reply back to the Hacker with the response or the output of the executed command. As this module runs with System right, it can execute any command. So Hacker has full control over the computer.

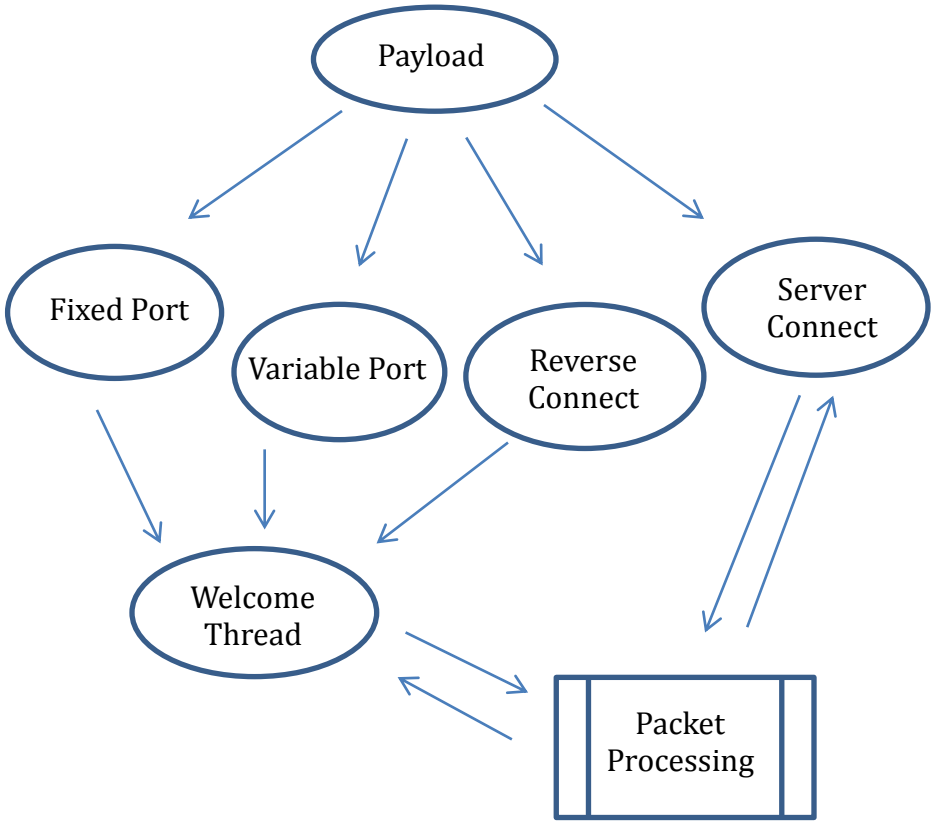


Fig. 8: Payload Module Workflow

Payload has 4 ways with which it can be controlled. Fixed Port thread listens on a particular port number, variable port thread listens on a randomly created port, reverse connect thread connects back to the hacker in case he is unable to connect directly and server connect thread lets the hacker control the Trojan via webserver.

In the first 3 cases, once the connection is established, a new thread is started. The objective of that thread is to receive commands, process it and sends the response back to the hacker. Commands from server connect thread gets directly processed and the response is updated in the website database.

In case a command is received by this module, it is executed and the output is stored in a file. This file is then transferred back to the Client. The Client then reads the file and displays the output in a correct format.

It also has a plugin facility where a function is loaded from a .dll library file and is executed during run time. For example, when it needs to take a screenshot, it loads a library and takes screenshot and unloads it. This way the size of the core remains small and function can be easily modified by replacing the screenshot library file.

Chapter 5: Database Structures

Fig.9: Table Name - Victims

```
C:\xampp\mysql\bin>mysql -u root daenerys
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 28
Server version: 5.1.41 Source distribution

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> describe victims;
+--------------+--------------+------+-----+---------+----------------+
| Field        | Type         | Null | Key | Default | Extra          |
+--------------+--------------+------+-----+---------+----------------+
| victim_id    | bigint(15)   | NO   | PRI | NULL    | auto_increment |
| OS           | varchar(100) | YES  |     | NULL    |                |
| version      | varchar(100) | YES  |     | NULL    |                |
| host_name    | varchar(100) | YES  |     | NULL    |                |
| login_name   | varchar(100) | YES  |     | NULL    |                |
| extrn_ip     | varchar(20)  | YES  |     | NULL    |                |
| ip_string    | varchar(100) | YES  |     | NULL    |                |
| lisn_port    | varchar(10)  | YES  |     | NULL    |                |
| reg_date     | varchar(50)  | YES  |     | NULL    |                |
| last_connect | varchar(50)  | YES  |     | NULL    |                |
| online       | int(1)       | YES  |     | 0       |                |
| safe         | varchar(2)   | YES  |     | NULL    |                |
+--------------+--------------+------+-----+---------+----------------+
12 rows in set (0.00 sec)
```

This table is used by Tracker module to store Victim's information. This helps the Hacker keep track of the Victim's IP address even if he/she has a dynamic IP address.

Fig. 10: Table Name – Flags

```
mysql> describe flags;
+--------------------+-------------+------+-----+---------+-------+
| Field              | Type        | Null | Key | Default | Extra |
+--------------------+-------------+------+-----+---------+-------+
| victim_id          | bigint(15)  | NO   |     | NULL    |       |
| sleep              | varchar(30) | NO   |     | 0       |       |
| self_destruct      | varchar(30) | NO   |     | 0       |       |
| key_logging        | varchar(30) | NO   |     | 0       |       |
| screen_logging     | varchar(30) | NO   |     | 0       |       |
| show_hacked_screen | varchar(30) | NO   |     | 0       |       |
| download_update    | varchar(30) | NO   |     | 0       |       |
| reverse_connect    | varchar(30) | NO   |     | 0       |       |
| update_conf        | varchar(30) | NO   |     | 0       |       |
| show_blue_screen   | varchar(30) | NO   |     | 0       |       |
| update             | varchar(3)  | NO   |     | 0       |       |
+--------------------+-------------+------+-----+---------+-------+
11 rows in set (0.00 sec)
```

These variables are used to control the actions of the Trojan. Server Connect Thread in Payload Module checks this table every 2 minutes for new commands. Once a command is received and actions are performed, this thread now updates the entry as *Executed*.

Fig. 11: Table Name – AdapterInfo

```
mysql> describe adapterinfo;
+-----------+--------------+------+-----+---------+-------+
| Field     | Type         | Null | Key | Default | Extra |
+-----------+--------------+------+-----+---------+-------+
| victim_id | bigint(15)   | YES  |     | NULL    |       |
| adp_index | int(3)       | NO   |     | 0       |       |
| adp_name  | varchar(100) | YES  |     | NULL    |       |
| adp_desc  | varchar(100) | YES  |     | NULL    |       |
| mac_addr  | varchar(20)  | YES  |     | NULL    |       |
| ip_addr   | varchar(16)  | YES  |     | NULL    |       |
| ip_mask   | varchar(16)  | YES  |     | NULL    |       |
+-----------+--------------+------+-----+---------+-------+
7 rows in set (0.01 sec)
```

- Stores all the network adapter info of the computer. Helps in determining MAC address and IP address before connecting to victims.
- IP_MASK helps in determining if the victim is in a large network or just a private network.
- Multiple adapter info helps in selecting the best way to connect. If the Victim can be connected via LAN, why to choose to connect over internet?

Fig. 12: Table Name – Config

```
mysql> describe config;
+-------------+-------------+------+-----+---------+-------+
| Field       | Type        | Null | Key | Default | Extra |
+-------------+-------------+------+-----+---------+-------+
| reverse_ip  | varchar(20) | NO   |     | NULL    |       |
| listen_port | varchar(10) | NO   |     | NULL    |       |
+-------------+-------------+------+-----+---------+-------+
2 rows in set (0.01 sec)
```

- Used to set the IP address and port number of the client for Trojans to reverse connect to it. So even if Hacker is on a move, he can update his IP address anytime and Trojan will connect to his new IP address.
- This table can also be used to send more configuration information in future updates.

Chapter 6: Conclusion

This design analysis and improvements over the traditional Trojan designing actually proved to be very successful. The Trojan evaded every detection mechanism by some of the best Anti-Viruses like Kaspersky, Bitdefender, etc. It was even tested in a network with a strict firewall along with a NAT server. The firewall allowed a very few ports like 80, 443, 2082, etc. This Trojan was able to bypass the firewall very easily and reverse connection was also tested and found to be working perfectly. Even when tested on a very large number of people, it remained undetected by any user. Size of the files could be kept less than 50KB each and the CPU consumption by each module was less than 1 MB.

This project was taken up for certain reasons. Few are:

- To aware Anti-Virus companies and Microsoft that their own facilities can be manipulated to bypass their own security.
- To emphasize that Microsoft should not provide so much access to the developers via APIs.
- To aware users that no Anti-Virus or Firewall can ever protect them from being hacked or infected by a well-designed Virus.
- To aware users not to trust any .exe file taken from anywhere except the genuine websites.
- To prove that criminals will always stay 1 step ahead of the security systems …

REFERENCES:

1. *Ghossoon. M. W. Al-Saadoon, Hilal M.Y. Al-Bayatti. A Comparison of Trojan Virus Behavior in Linux and Windows Operating Systems, WCSIT, Vol. 1, No. 3, 56-62, 2011*
2. *Aelphaeis Mangarae. Trojan White Paper, Igniteds Security Community 2006*
3. *Comer,Douglas E. Internetworking with Tcp/Ip: Principles, Protocols, and Architectures, Ed. 4, New Delhi, Preitice Hall, 2002*
4. *Fadia, Ankit. Network Security: A Hacker's Perspective, Ed. 2, Delhi, McMillan, 2002*
5. *Weaver, Randy. Guide To Network Defence And Countermeasures, Australia, Course Technology, 2007*
6. *Ohlund, Jim. Network Programming For Microsoft Windows, Bangalore, WP Publisher, 1999*
7. *Petzold, Charles. Programming Windows , Ed. 5, Bangalore, WP Publisher, 1999*
8. *Wells, Nicholas. Guide To Linux Networking And Security, Australia, Course Technology, 2003*
9. *Ciampa, Mark. Security+ Guide To Network Security Fundamentals, Ed. 2, Australia, Course Technology, 2005*
10. *Dunn, Kyle. C++ Application From Inspiration To Implementation, Navi Mumbai, SHROFF, 2003*